Low-Cost Hardware Infrastructure for Runtime Thread Level Energy Accounting

Marius Marcu, Oana Boncalo, Madalin Ghenea, Alexandru Amaricai, Cosmin Cernazanu
University Politehnica Timisoara
Timisoara, 2 Vasile Parvan Blvd.
mmarcu@cs.upt.ro, oana.boncalo@cs.upt.ro, alexandru.amaricai@cs.upt.ro, cosmin.cernazanu@cs.upt.ro

Jan Weinstock, Zheng Wang, Rainer Leupers
RWTH Aachen, Germany
weinstoc@ice.rwth-aachen.de, wang@umic.rwth-aachen.de, leupers@ice.rwth-aachen.de

ABSTRACT
The ever-growing need for energy efficient computation requires adequate support for energy-aware thread scheduling that offers insight into a systems behavior for improved application energy/performance optimizations. Runtime accurate monitoring of energy consumed by every component of a multi-core embedded system is an important feature to be considered for future designs. Although, important steps have been made in this direction, the problem of distributing energy consumptions among threads executed on different cores for shared components remains an ongoing struggle. We aim at designing a generic low-cost and energy efficient hardware infrastructure which supports thread level energy consumption monitoring of hardware components in a multi-core system. The proposed infrastructure provides upper layers (operating system and application threads) with per thread and per component energy accounting API (Application Programming Interface), similar with performance profiling functions. Implementation results indicate that the proposed LEM (Load and Energy Monitor) adds a less than 3% resource overhead to the monitored system.

Categories and Subject Descriptors
C.1.4 [**Processor architectures**]: Parallel Architectures – *Distributed architectures.*

General Terms
Measurement, Design.

Keywords
Energy accounting, power consumption, runtime monitoring, heterogeneous multi-core systems, Per Thread Energy Accounting (PTEA).

1. INTRODUCTION
Energy metering has been a major research topic during the last years. Measuring energy is needed to validate and calibrate energy models; to perform energy profiling of hardware and software applications and, last but not the least, to develop energy-aware applications based on runtime energy measurement. Based on their intrusiveness and required hardware support, these techniques span from software approaches [1], to solutions requiring dedicated hardware support within the system such as a customized token ring interconnect [2], network on a chip interconnect with monitors [3][4]. Approaches such as HEMA (Hardware-assisted energy monitoring architecture) [5], are a tradeoff that rely on software techniques and usage patterns, with some periodic calibration from hardware monitors. These researches try to distribute power consumption per core. However, as argued in the work of [6], this is no longer sufficient. Per task metering (PTEM) [6] is a must in nowadays systems in order to facilitate: efficient resource allocation for task execution (dynamically assess at operating system (OS) level resource allocation), system level energy/performance optimization, billing in datacenters.

Furthermore, breaking down energy consumption is even harder, and requires support on both the monitoring side, and on the OS side. The majority of the previous work has been focused on breaking the system energy or power consumption down to component level, using power measurements or power estimations. Thus, the problem of per component energy accounting is well studied at the moment, with well understood constraints arising due to the limited time resolution and/or accuracy of the power sensors. However, the ever-increasing complexity of systems and system-level interactions, requires Per Thread Energy Accounting (PTEA), also referred to as PTEM in the work of [7], bringing energy metering to a higher abstraction level. PTEA performs energy estimation of the hardware components in response to the actions initiated by each specific task in a multi-tasking environment. Task level and thread level energy accounting techniques (PTEA) are even more complex to accomplish because they have to split power consumed by shared components to the tasks or threads that control them. This is only possible with dedicated hardware and software support.

The contribution of this work is as follows:

- Infrastructure for dynamic energy consumption monitoring in a heterogeneous multi-core system with per thread energy accounting (PTEA);
- A use case on the software side (OS and drivers) for run-time PTEA implementation.

Briefly, PTEA can be achieved splitting the whole energy into processing energy (energy consumed by processing cores), data movement energy (energy consumed by interconnects to read and store data) and data storage energy (energy consumed by memories to store task data). The proposed infrastructure addresses: processing energy accounting, data movement energy accounting, and data storage accounting. Both processing and data movement accounting is performed per thread, but data storage is currently per component. However, data storage energy splitting per task can be further implemented in our infrastructure using the techniques proposed by [6].

The PTEA infrastructure is based on centralized-distributed-interconnect architecture as described in Section 3. The features of the proposed solution are:

- Wishbone interconnect: an open-source standard, with one proposed addition: a short bus cycle when no relevant data can be read by monitoring infrastructure;
- Centralized control with a number of configurable parameters that allow an easy adaptation of the infrastructure for various systems and use cases;
- Lightweight non-intrusive sensors attached to processing and shared system's components (see description from Section 3);
- Software driver support for programming the infrastructure; it should provide the needed support for PTEA at the OS side.

This paper is organized as follows: Section 2 provides the description of previous related work on energy accounting; Section 3 presents the proposed energy accounting infrastructure; Section 4 describes proposed PTEA solution; experimental results and system overhead is presented in Section 5. The conclusions are discussed in the last section.

2. RELATED WORK

Hardware monitoring infrastructures have been proposed in the past, with results reported for both FPGAs and ASICs. These solutions have to provide several main features: power and performance meters (physical or model based sensors), data collection (interconnects), control unit (configuration and processing) and finally, software drivers (API).

The work presented in [2] uses a customized token ring interconnect for monitoring and actuation. Two lines are dedicated for communication control (token and valid), with as many as desired data lines possible (based on the transmission speed and/or resources committed for the infrastructure). This solution is simple, with a limitation for the transmission delay: it increases linearly with the number of nodes. Research in [3][4], relies on a network on a chip (NoC). Hardcoded routing tables and two types of monitors (i.e. data pull and data push) try to limit the overhead introduced by the NoC. Another solution, proposes a light-weight monitoring system relying on a single-wire interconnect network, where monitoring components take turns to send data to centralize data [7].

Most of the solutions allow centralized control [1-7], with [2] also permitting distributed control and information aggregation. When addressing large platforms, these are broken into subsystems (in some works referred to as islands), with each subsystem having its own interconnect and having its own monitoring infrastructure. For example, for large systems, the authors of [2] divided the SoC on islands and use a customized ring for each island. Inter-subsystem messages can be exchanged by means of an interconnect on top of the aforementioned one [2][3][4]. This type of solution is also adopted in this work.

Sensor/monitor architecture is typically made of two parts: interface with the choice of interconnect, and processing part, where the data is being aggregated/actuated (sensing application dependent). Furthermore, on the processing side (which is connected to the system being monitored), the coupling between the sensors and the target system can be classified in two categories: tight (when actual instrumentation of the component design is required for introducing event counters and additional ports for reporting their activity [6]), and weak (when already built-in support exists – e.g. thermal sensors, performance counters, or some sort of non-intrusive way of accounting events is proposed [3][4][7]).

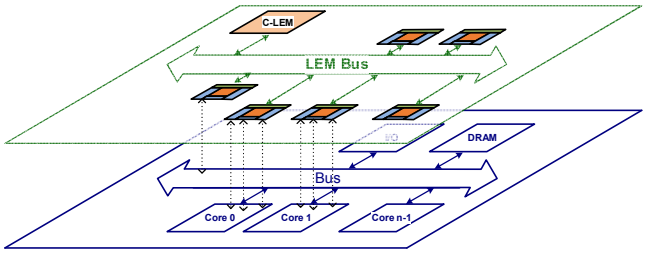

Figure 1. Overall solution architecture.

PTEM is discussed only by few papers [6][8], most of them targeting x86 processor system architectures [8]. In [6] PTEM is achieved on the basis of resource utilization and occupancy tracking with dedicated hardware. Considering the state of the art, most of the recent efforts in energy accounting are directed toward component level energy modelling, profiling, or monitoring. However, energy consumed by data movement is not taken in consideration, due to its complexity and shared usage problem. In our work we address energy accounting of data movement in a heterogeneous multi-core system in conjunction with energy accounting of processing elements.

3. PTEA INFRASTRUCTURE

3.1 Overall solution architecture

The proposed solution, called Load and Energy Monitor (LEM), implements a distributed sensor network through a dedicated bus interconnect. The control is centralized. LEM infrastructure is constructed on three components: LEM sensors (S-LEM), LEM controller (C-LEM) and LEM interconnect (LEM Bus) (see Figure 1). Each component of the target system needs to implement its own accounting sensors which will provide LEM interconnect with access interface to collect sensors data. The sensors (S-LEM) collect data directly from hardware, perform component level performance or energy accounting computations and allow fine tuning of components' power management parameters. This is a generic infrastructure, therefore sensors sampled values may be switching activity, performance counters, and power or temperature physical measures. Sensors are attached to processing cores, interconnects and memory components of the target monitored system. Each component should implement at least one sensor – the energy accounting sensor. The LEM infrastructure is also non-intrusive, in the sense that the monitoring infrastructure plane is decoupled from the monitored target functional system (Fig. 1). Decoupling is possible by providing sensors with standard bus interface to LEM bus and data collection probes to monitored hardware and/or bus monitor. The LEM bus based on Wishbone Bus (WB) specification, is cost-effective and light-weight.

3.2 Hardware Components Description

The LEM sensors implement a standard interface for the various performance and/or consumption monitoring support existing in hardware. Every sensor is made of three parts (Fig. 2): (1) sensor interface that is Wishbone (WB bus) based it is used to connect the sensor core with the central LEM; (2) sensor back-end (with

some processing and tuning support) with limited energy consumption accounting and (3) hardware interface probes which is hardware or bus specific.

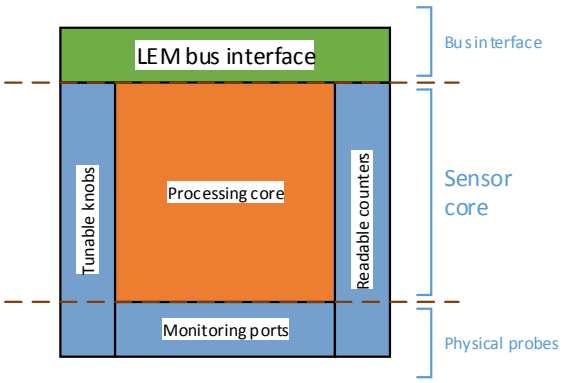

Figure 2. Overall sensor architecture.

Sensor back-end either reads the energy consumed by a component (some processing cores offer this information through some dedicated ports), or derives the energy consumed by adding for each accounted event a value from a stored table with *apriori* computed values. Although the target system is different, some common points exist with the tool HEMA [5], mainly the idea of reducing the overhead on the hardware monitoring infrastructure. Our solution tries to derive energy consumption by correlating event accounting (i.e. event counters at various levels in our system) with hardware based energy pre-computed luck-up-tables.

LEM sensors can be configured to provide energy, power or raw data. Energy accounting is performed by summing up the samples from the hardware until then next poll from the master C-LEM. Power monitoring is performed by averaging the samples from the hardware until the next read from the master. Raw data from low-level hardware monitors are passed directly as instantaneously values to the master. Using monitoring ports, LEM sensors can access underlying monitoring hardware (e.g. performance counters, physical sensors, or events monitors).

Central LEM (C-LEM) controls and collects sensors data from installed LEM sensors and provides the OS with structured access to the sensors' data (Fig. 3). The OS communicates with the central LEM unit through the global memory address space for sensor control and sampling. Data received from the LEM sensors is stored in the C-LEM local memory. Each sensor has a reserved 128 memory structure as presented in Fig. 3. The stored information represents the instantaneous or aggregated value of the data read from the respective sensor. Each memory location will store the last value calculated from all the sensors that have sensor data corresponding to the current core.

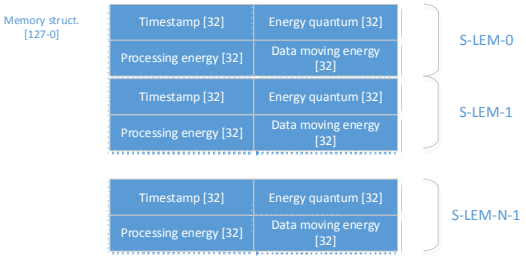

Figure 3. LEM memory

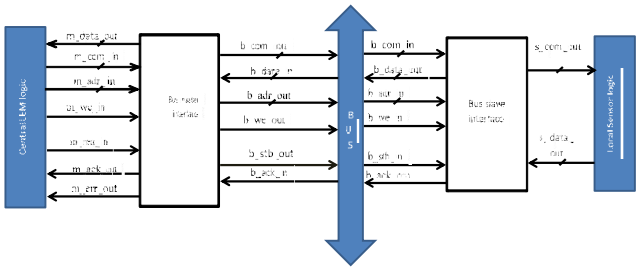

Figure 4. LEM interconnect

Furthermore, C-LEM central is also highly parameterized, and it can be programmed by the underlying software same as the router from [3]. Dedicated registers allow the programming of the LEM for: enabling/disabling monitoring for a component, changing the sampling rate, controlling the power model or selecting the accounting mode. LEM is a bus based infrastructure, based on WB standard, with the master and slave interfaces presented in Fig. 4. The bus supports one master, which is C-LEM, and many sensors slaves. In the initial version of LEM *256* sensors (S-LEM) are targeted. The C-LEM master polls at the configured rate all enabled sensors and executes non burst reads to get sensors' values.

4. PER THREAD ENERGY ACCOUNTING

LEM infrastructure is a hardware solution that can be customized for different applications. It has been further customized and used to implement PTEA in a multi-core system. In a multi-core system we consider two types of components: processing cores (could be homogeneous or heterogeneous) and shared resources (e.g. memories and interconnects). In a multi-core hardware environment, every core has a unique core ID. Our proposal is to use this ID in hardware transaction with shared components to identify the processing core which will be charged with the energy budget of the current transaction. The LEM sensors connected to these shared components will use master ID to account per core energy consumption.

Modern interconnects like Wishbone (open source) and AXI (ARM) allow system designers to attach meta-information to each transfer. For example, Wishbone bus standard specifies user-defined tags to apply extra information to each bus cycle [9]. On the other hand, AMBA open specification [10] associates implicitly hardware IDs of the master to each bus cycle. ARID, RID, AWID, WID and BID bus signals are carrying ID tags of the read address, read, write address, write and response bus transfers. Hence, it makes sense for the LEM sensors of shared resources to use the master ID of the access to split the energy consumed for shared components.

Using the LEM infrastructure and the ID tagging bus support, PTEA implementation can be split in two steps:

- Per core energy accounting of processing cores and shared resources based on hardware support;
- Per thread energy accounting implemented at OS level during context switching, using the provided LEM drivers.

LEM driver provides OS with an API similar to existing performance profiling counters functions. The LEM driver interface is based on start/stop accounting operations.

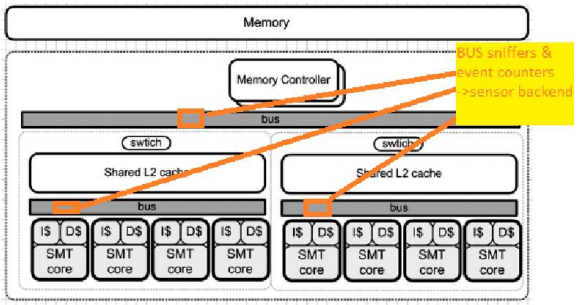

Figure 5. Memory hierarchy monitoring (adapted from [6])

When a thread context switch occurs, the OS will store the energy counters of the current thread and will restart the counters for the next thread. While a thread is executed by a core, per core energy accounting implemented in hardware will be used to account for the energy of a running thread. Considering that a core will execute only one thread at the time, the coordination between OS and HW will account for the thread level energy in a multi-core/multi-threading execution environment.

LEM hardware infrastructure comprises of bus/interconnect sniffers (S-LEM). These monitor for the number and type of accesses for each shared components. Energy accounting is estimated using look-up-tables which correlated the energy consumed with the number and type of accesses. We further discuss the PTEA use case for the example target platform used in [6] - Figure 5. In case *core0*, from *cluster0*, has a cache miss in *L2 - cache*, three events are reported by three sensors: access *L1 - cache* (by the processing core's sensor), access *L2 - cache* (event reported by the cluster bus: L2 cache access), access memory controller (event reported by the system bus). Each of these events are accounted on behalf of *core0*. The proposed approach differs from the one described in [6] by the fact that events such as dirty line, cache line eviction, cache hit or miss are not monitored solely on the shared component side. These are a direct consequence of our proposed accounting method, and are transparent for the infrastructure. For example, a cache line eviction require a number of events on the system bus; the shared cache sensor for the processing core (identified by master ID) is notified by the sensor on the system bus that energy needs to be accounted on its behalf. This mechanism is hierarchical. Furthermore, the means for these types of notifications represent transactions on the LEM infrastructure. Therefore, the bus topology of the LEM infrastructure is similar with the one in the monitored system.

Table 1 – LEM Cost Estimates and Overhead

Resource type	**LEM resources**	**MPSoC resources**	**Overhead [%]**
Slice LUTs	750	24063	3
Slice registers	742	40544	1.8
BRAM	0.5	104.5	0.05
DSP	0	12	0

5. EXPERIMENTAL RESULTS

The LEM infrastructure has been implemented on the Xilinx ZC702 evaluation kit, with Xilinx Zynq-7020 device. The Zynq 7000 family of devices combine two ARM Cortex A9 cores with FPGA fabric in one MPSoC. The development board has built-in power monitoring sensors for the power lines of the main components: processing system cores (ARM cores), programmable logic core, DRAM, I/O etc. The reference design for PTEA presented in Figure 10 has four Microblaze cores with local interrupt controller, and local instruction and data memory (Figure 7).

The implementetion overhead of the LEM infrastructure for the reference design is presented in Table 1. C-LEM samples the sensors at configurable rates. One polling cycle is presented in Figure 6. LEM bus signals are derived from WB standard. Minimum delay of one polling cycle function of the number of available LEM sensors shows a linear dependence (Figure 8).

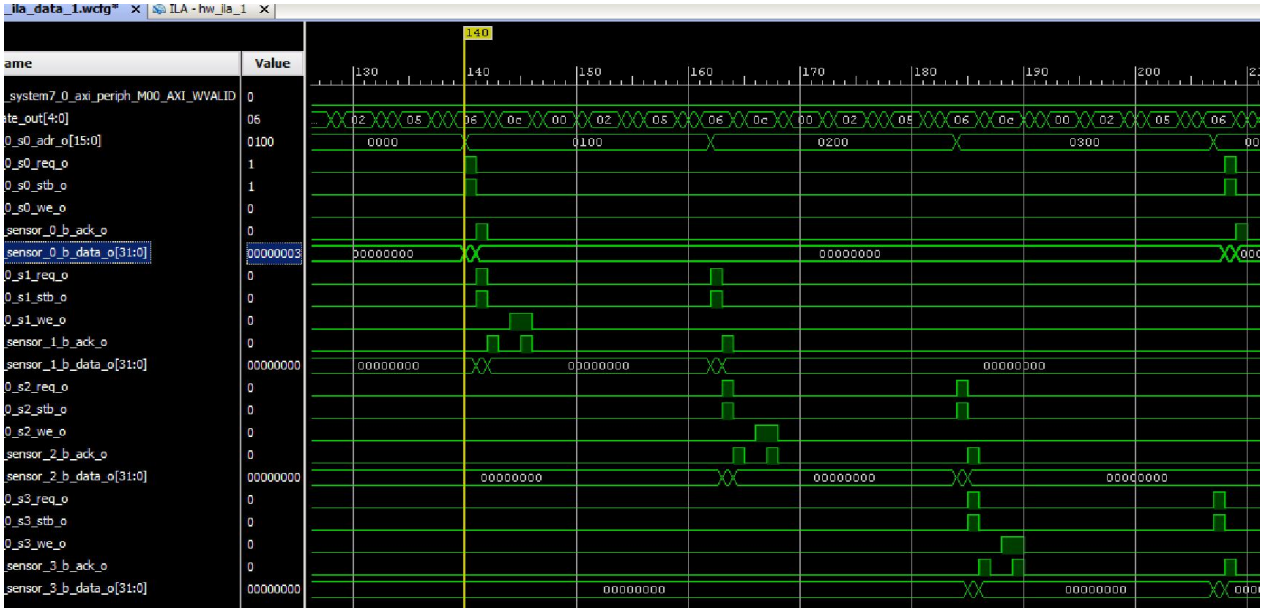

Figure 6. LEM polling cycles

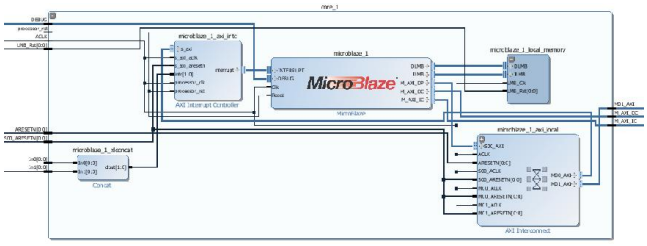

Figure 7. Power consumption of Microblaze instructions

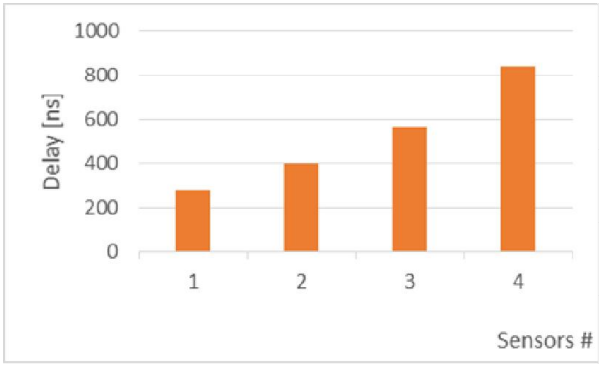

Figure 8. LEM polling time vs. number of sensors

Preliminary instruction level profiling using the LEM implementation for the reference design is presented in Figure 9. The sensors monitor continuously the component interfaces, mapping the current operation onto the corresponding power consumption value, using pre-computed tables. The sensor of shared components (e.g. main memory) monitors memory transfers on AXI bus and dispatches the measurement values based on the core ID transferred on AXI ID lines. The LEM sensors has been calibrated using built-in power sensors and XADC controller within the Zynq.

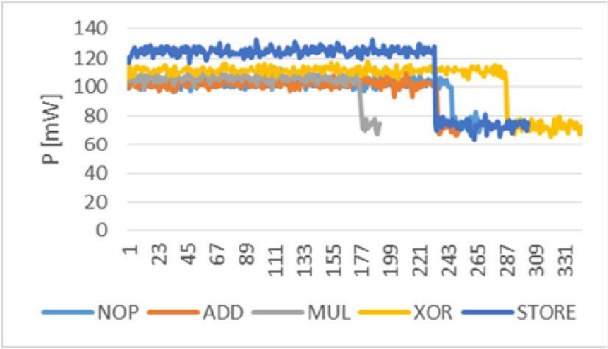

Figure 9. Power consumption of Microblaze instructions

6. CONCLUSIONS

In this paper, we have introduced a cost effective LEM infrastructure for component level power and energy monitoring. The monitoring infrastructure implements two levels of energy accounting: processing energy and data movement energy. Per core energy accounting can be done using the LEM hardware infrastructure. The infrastructure can be further used in conjunction with OS drivers, to implement thread-level energy accounting. The most important limitation for the proposed infrastructure is represented by the linear increase of the polling time with the number of sensors. We will address this limitation by shortening bus cycle when no relevant data can be read by the monitoring infrastructure. Further tests cases on the reference design have to be performed in order to validate the PTEA.

7. ACKNOWLEDGMENTS

This work has been supported by the project CHIST-ERA/1/01.10.2012 – "GEMSCLAIM: GreenEr Mobile Systems by Cross LAyer Integrated energy Management".

8. REFERENCES

[1] Weaver, V. M., Johnson, M., Kasichayanula, K., Ralph, J., Luszczek, P., Terpstra, D., and Moore, S., 2012. Measuring Energy and Power with PAPI. *Proceedings of 41st International Conference on Parallel Processing Workshops* (Pittsburgh, PA, USA, September 2012). ICPPW '12.

[2] Bouajila, A., Lakhtel, A., Zeppenfeld, J., Stechele, W., Herkersdorf, A., 2012. A low-overhead Monitoring Ring Interconnect for MPSoC Parameter Optimization. *Proceedings of IEEE International Symposium on Design and Diagnostics of Electronic Circuits & Systems* (Tallinn, Estonia, April, 2012). DDECS 2012.

[3] Madduri, S., Vadlamani, R., Burleson, W., and Tessier, R., 2009. A Monitor Interconnect and Support Subsystem for Multicore Processors. *Proceedings of Design, Automation & Test Europe* (Nice, France, April 2009). DATE 2009.

[4] Zhao, J., Madduri, S., Vadlamani, R., Burleson, W., and Tessier, R., 2011. A Dedicated Monitoring Infrastructure for Multicore Processors. *IEEE Transaction on Very Large Scale Integration (VLSI) Systems*, Vol. 19, No. 6, (June 2011).

[5] Choi, S., Hwang, H., Song, B., and Cha, H., 2012. Hardware-assisted energy monitoring architecture for micro sensor nodes. *Journal on System Architecture*, Elsevier, No 58, (2012), pp 73-85.

[6] Lui, Q., Moreto, M., Jimenez, V., Abella, J., Cazorla, F.J., Valero, M. 2013. Hardware Support for Accurate Per-Task Energy Metering in Multicore Systems. *ACM Transactions on Architecture and Code Optimization*, Vol. 10, No. 4, (December 2013).

[7] Ituero, P., López-Vallejo, M., Marcos, M. A. S., and Osuna, C. G., 2012. Light-Weight On-Chip Monitoring Network for Dynamic Adaptation and Calibration. *IEEE Sensors Journal*, Vol. 12, No. 6 (June 2012).

[8] Molka, D., Hackenberg, D., Schone, R., and Millier M.S. 2010. Characterizing the Energy Consumption of Data Transfers and Arithmetic Operations on x86-64 Processors. In *Proceedings of International Green Computing Conference* (Chicago, USA, August 15-18, 2010). GreenComp 2010. IEEE, 123-133. DOI: 10.1109/GREENCOMP.2010.5598316

[9] OpenCores, WISHBONE System-on-Chip (SoC) Interconnection, 2010.

[10] ARM, AMBA Open Specifications, http://www.arm.com/products/system-ip/amba/amba-open-specifications.php, 2003.

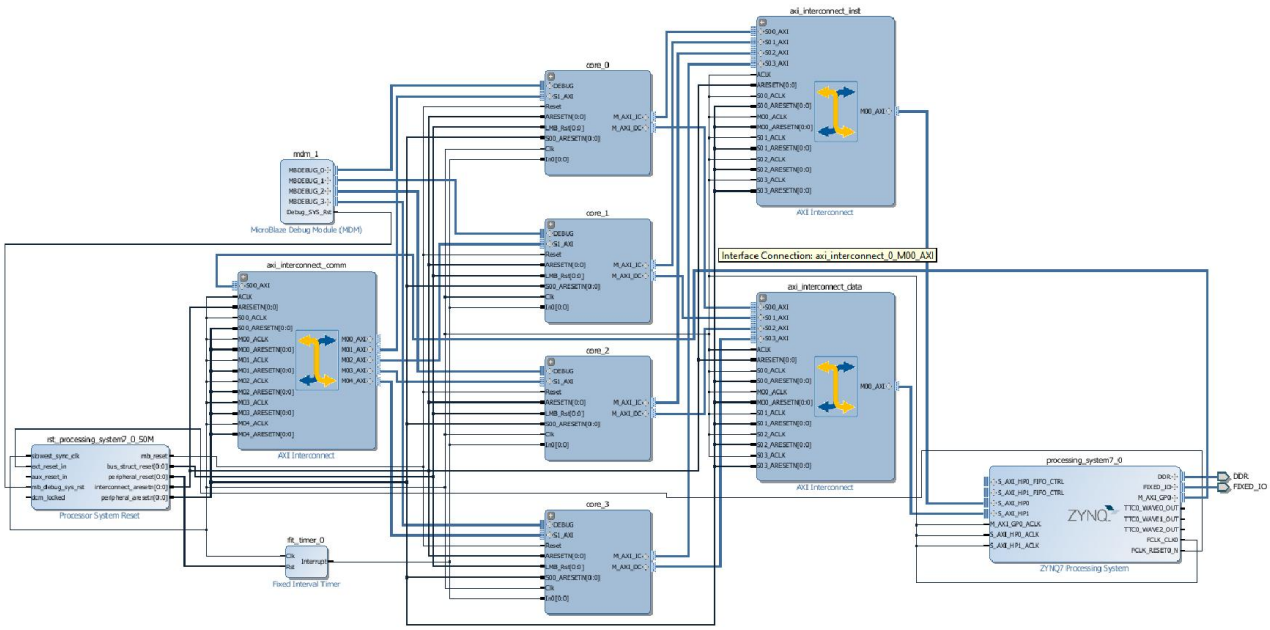

Figure 10. Reference system design